CS3723 Python Pgm 6 (70 pts) Python – due 12/3/2018

This is a continuation of Program 5. In this assignment, you will execute a BEEP program and show its output.

In Progam 5:

- Placed locations of labels in a dictionary.
- For VAR statements, placed variable names and values in dictionaries.
- Printed variable and label informaiton.

In Program 6:

1. Execute each BEEP statement.
    - VAR - call your function from program 5.
    - ASSIGN – assign the value of the expression to the specified variable which must have been declared
    - IF – if the expression is True, branch to the specified label. Otherwise, continue with the next statement.
    - PRINT – print each of the specified variables or literals
    - GOTO – branch to the specified label
2. Begin execution by printing the message:
    ```
    execution begins ...
    ```
3. Handle the optional **-v** switch to the command:
    ```
    python3 p6Driver.py p6Input.txt
    python3 p6Driver.py p6Input.txt -v
    ```
    Specifying the **-v** causes your execution to be verbose:
    - for each line being executed, print a message like the following
      ```
      executing line 10: ASSIGN count + count 1
      ```
4. Your code should **raise** several **exceptions** which should be caught by your execution function and cause the program to terminate:
    ```
    TooFewOperands - the various operations were given too few operands (e.g., only one
          operand for a greater than)
    VarNotDefined - a referenced variable is not defined
    LabelNotDefined - a referenced label is not defined
    InvalidExpression - other problems with expressions such as unknown operator
    InvalidValueType - an operation expecting an INT had a value which was of the wrong type
    You may add additional exceptions if needed.
    ```
    Example raising an exception:
    ```
    raise InvalidValueType("'%s' is not numeric" % (op1))
    ```
    Example exception class:
    ```
    class InvalidValueType(Exception):
        def __init__(self, *args, **kwargs):
            super().__init__(self, *args, **kwargs)
    ```
    Example exception handler which references the global variable `lineNum`:
    ```
    try:
        some Python code
    except (InvalidValueType, other exception classes) as e:
        print ("*** line %d error detected ***" % (lineNum))
        print("%-10s %d *** %s ***" % (" ", lineNum, str(e.args[1])))
        break
    except Exception as e:
        print("*** line %d error detected ***" % (lineNum))
        print(e)
        break
    except:
        print("*** line %d error detected ***" % (lineNum))
        traceback.print_exc()
        break
    ```
5. It would be wise to include a statement execution limit of perhaps 5,000. Within your execution function, increment a limit counter for each statement executed. If it reaches 5,000 statements, terminate and show an

error that an infinite loop was most likely encountered. After execution completes, print the message "**execution ends, *xx* lines executed**" where *xx* is the number of lines executed.

6. Labels at the beginning of lines are NOT redefined when **executing**. Simply ignore those labels during execution of BEEP code.
7. I have provided the following Python function as an example:

```
def evalGreater( op1, op2):
    try:
        iVal1 = int(op1)
    except:
        raise InvalidValueType("'%s' is not numeric" % (op1))
    try:
        iVal2 = int(op2)
    except:
        raise InvalidValueType("'%s' is not numeric" % (op2))
    return iVal1 > iVal2
```

Note that its caller probably recognized that one of many operations of two operands was being called. It probably determined what each operand was :

a character literal – "string" and removed the beginning and ending "

a numeric constant – the operand is a numeric string until the operation converts it

a variable reference – used the value of the variable for the operand

BEEP statements

VAR *dataType variableName initialValue*

This declares the specified *variableName* to be of the specified data type. Use the *variableName* as the key in both the varTypeD and varValueD dictionaries. If specified, the *initialValue* is the initial value for the variable *variableName*. Examples:

```
VAR int count 0
VAR string name
VAR string greeting "Hello"
```

# *comment*

label*: *statement

The subscript for this line of code is stored in the labelD dictionary.

***statement* is one of:**

ASSIGN *variableName expression*
IF *expression label*
PRINT *varLiteral1 varLiteral2 …*
GOTO *label*

expression is one of:

varLiteral	return the string value (without the ") for a string literal, the value of a numeric constant or return the value of a variable
* *varLiteral varNumber*	return a string with *varLiteral* replicated *varNumber* times
+ *varNumber1 varNumber2*	return the sum of the values
- *varNumber1 varNumber2*	return the difference of the values (*varNumber1* minus *varNumber2*)
> *varNumber1 varNumber2*	return True if *varNumber1* > *varNumber2;* this is a numeric comparison
>= *varNumber1 varNumber2*	return True if *varNumber1* >= *varNumber2;* this is a numeric comparison
& *varLiteral1 varLiteral2*	return the concatenation of the two strings

Program Requirements

1. You will be provided with multiple input files. Your program should be passed the name of one file as a command argument. Also see the optional `-v` switch. Try your program on each of the sample BEEP program files.
2. Your Python source code must be separated into **multiple functions** and **source files**:

   p6Driver.py This is the main driver for your code. The program is passed the name of the BEEP source code file an an optional verbose flag. It reads and prints all the BEEP source code lines and places them in a list.

   It does the remaining requirements done in program 5.

   It should then invoke your execution function which should be in p6Exec.py.

   p5Dict.py This code provides the **declareVar, printVariables,** and **printLabels** functions**.** It can have other functions if needed.

   p6Exec.py This code provides the ability to execute BEEP source code. It will contain the bulk of the functions for program 6 unless doing extra credit.

   To import code from another file, use a Python statement like this:

   ```
   from fileName import funcName1, funcName2
   ```

   Note the actual file would be named *fileName.*py. Inotherwords, the filename specified on the **from** would be without the `.py`.

3. If the same variable name is encountered in multiple VAR statements, simply re-declare the variable with its new type and possibly value. Variable names in BEEP are **not** case sensitive.
4. If the same label is encountered at the beginning of two or more statements, show an error message (see below) and the label. Labels are **not** case sensitive. Note that labels at the front of BEEP source code lines are **ignored** during execution. If a label appears on multiple lines, print a message showing its first line number and the other one:

   ```
   ***Error: label 'LOOP' appears on multiple lines: 9 and 20
   ```

5. Documentation:
   - All your code should be documented.
6. **Any use of code from another web site will result in a 0 on this assignment, may result in an F for this course, and may cause you to be expelled from UTSA.**
7. Execute your code on a fox server using **python3**
8. Turn in a zip file named *LastnameFirstname*.zip which contains:

   p6Driver.py
   p5Dict.py
   p6Exec.py (if doing the extra credit, instead you will provide Executor.py)
   p6Out.txt – contains the output generated by your program for the **p6InputC.txt** input file

   Do not include a directory in your zip file.

Extra Credit (3 points + 120 / n) – using an Executor object

- **Late** submissions will **not** receive extra credit.
- Instead of most of the code being in p6Exec.py, your code for executing will use an Executor object. If you want to use additional classes, please include their source code in your submission.
- **Executor** object:
  - uses an **instance variable** for the current line number instead of a global variable
  - uses **instance variables** for each of the dictionaries instead of passing them to the functions
  - its constructor is passed the list of BEEP source code statements and dictionaries for labels, variable types, and variable values
  - your code must not have any global variables
- Your code must properly handle all cases in all input files to be eligible for extra credit.

Sample Input:

```
# p6InputA.txt
# © 2018 Larry W. Clark, this document may not be copied to any other website.
VAR int count 0
VAR string result
VAR string symbol "ho"
VAR int tick 3
VAR int limit 10
VAR string greeting "hello..there"
VAR int iter 4
PRINT "begins..."
PRINT "Top:...count=" count "tick=" tick "symbol=" symbol
ASSIGN count + count 1
IF > tick limit pastlimit
    ASSIGN result * symbol tick
    PRINT result
    ASSIGN tick + tick 1
    GOTO afterIf
pastlimit: PRINT "***tick...reached...limit:" tick
afterIf: PRINT "Bot:...count=" count "tick=" tick "symbol=" symbol
PRINT "EndPgm"
```

Sample Output:

```
BEEP source code in p6InputA.txt:
  1. # p6InputA.txt
  2. # c 2018 Larry W. Clark, this document may not be copied to any other website.
  3. VAR int count 0
  4. VAR string result
  5. VAR string symbol "ho"
  6. VAR int tick 3
  7. VAR int limit 10
  8. VAR string greeting "hello..there"
  9. VAR int iter 4
 10. PRINT "begins..."
 11. PRINT "Top:...count=" count "tick=" tick "symbol=" symbol
 12. ASSIGN count + count 1
 13. IF > tick limit pastlimit
 14.     ASSIGN result * symbol tick
 15.     PRINT result
 16.     ASSIGN tick + tick 1
 17.     GOTO afterIf
 18. pastlimit: PRINT "***tick...reached...limit:" tick
 19. afterIf: PRINT "Bot:...count=" count "tick=" tick "symbol=" symbol
 20. PRINT "EndPgm"
Variables:
    Variable     Type     Value
    COUNT        INT      0
    GREETING     STRING   hello..there
    ITER         INT      4
    LIMIT        INT      10
    RESULT       STRING
    SYMBOL       STRING   ho
    TICK         INT      3
Labels:
    Label        Statement
    AFTERIF      19
    PASTLIMIT    18
execution begins ...
begins...
Top:...count= 0 tick= 3 symbol= ho
hohoho
Bot:...count= 1 tick= 4 symbol= ho
EndPgm
execution ends, 19 lines executed
```